Bayesian Decomposition of Multi-Modal Dynamical Systems for Reinforcement Learning

Markus Kaiser[a,b,*], Clemens Otte[a], Thomas A. Runkler[a,b], Carl Henrik Ek[c]

[a] *Siemens AG, Otto-Hahn-Ring 6, 81739 Munich, Germany*
[b] *Technical University of Munich, Boltzmannstraße 3, 85748 Garching, Germany*
[c] *University of Bristol, MVB, Woodland Road, Clifton BS8 1UB, United Kingdom*

Abstract

In this paper, we present a model-based reinforcement learning system where the transition model is treated in a Bayesian manner. The approach naturally lends itself to exploit expert knowledge by introducing priors to impose structure on the underlying learning task. The additional information introduced to the system means that we can learn from small amounts of data, recover an interpretable model and, importantly, provide predictions with an associated uncertainty. To show the benefits of the approach, we use a challenging data set where the dynamics of the underlying system exhibit both operational phase shifts and heteroscedastic noise. Comparing our model to NFQ and BNN+LV, we show how our approach yields human-interpretable insight about the underlying dynamics while also increasing data-efficiency.

1. Introduction

Machine learning methods [1] are designed to solve tasks where the underlying system we want to model is only partly known or understood. The hope when using machine learning methods is that we can reduce this uncertainty by exploiting statistical patterns in data generated from the underlying system.

Reinforcement learning (RL) [2] is a machine learning paradigm designed to learn in a dynamic environment where we can specify a goal or have a notion of what a desirable behaviour is. The goal of RL is to learn a policy which dynamically chooses actions in the environment in order to achieve a goal or a behaviour. Specifically, an RL agent's task is to learn a policy π which, given the current state $\boldsymbol{s}$ of an environment, chooses an action $\boldsymbol{a}$ to achieve the goal specified by a reward function r mapping system states to numerical rewards. To learn a policy, any RL system needs to understand the underlying dynamics governing the system, how the transition between states is effected by the actions

*Corresponding author
Email address: markus.kaiser@siemens.com (Markus Kaiser)
URL: mrksr.de (Markus Kaiser)

taken. The next state $\boldsymbol{s}'$ is determined by the latent and possibly stochastic transition function $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a})$. How the dynamical system is treated is one of the main distinctions among different approaches to RL. In model-based RL, the dynamic model is an explicit part of the system, while in the model-free counterpart the transition dynamics are implicit and cannot be disentangled from the system.

Applying RL in an industrial setting often implies trying to derive an alternative more efficient controller for an already existing system. In a critical application it is unlikely that we will be able to deploy an untested policy as this can lead to safety issues. This means that in practice we are often limited by previously collected data created by a different, possibly known, control mechanism in order to learn our model. In the literature, this scenario is referred to as batch RL [3], where we are presented with a set of state transitions $\mathcal{D} = \{(\boldsymbol{s}_n, \boldsymbol{a}_n, \boldsymbol{s}'_n)\}_{n=1}^{N}$ and are unable to interact with the original system to find a policy. In order to be able to derive an efficient policy in this scenario, we need to use the available data as efficiently as possible. Data efficiency in machine learning comes from reducing the search space of solutions. In other words, data efficiency arises from being able to exploit as much prior knowledge of the system as possible [1].

To be able to use the available data as efficiently as possible we therefore need a model which provides explicit and interpretable handles such that we can easily introduce priors. The model-based approach to RL describes each component of the system in a modular fashion thereby providing an interface to incorporate prior knowledge. The challenge is how these priors should be specified and how they should be included such that the hypothesis space can be limited in a manner coherent with our knowledge of the system.

Gaussian processes (GPs) are stochastic processes that can be used to specify probability distributions over the space of functions. While a GP specifies a distribution with support for all functions, it efficiently concentrates its probability mass to functions with specific characteristics. These characteristics make GPs well suited for RL as they do not impose hard constraints while still placing a significant structure on the space of functions. In [4] the authors propose a model-based RL method where Gaussian processes are used as priors for the dynamics. They provide a principled approach of taking model uncertainty into account when evaluating the performance of a policy, thereby reducing the impact of model-bias. However, the approach has several restrictions, transition dynamics are modelled as standard Gaussian processes (GPs) and policies and rewards must be of specific forms.

In this work, we will show how we can alleviate some of the limitations of [4] to provide a richer and more efficient RL model. We will show how we can introduce additional constraints on the dynamic model allowing for multiple transitional signatures to be active simultaneously. Incorporating this knowledge facilitates learning by allowing us to more precisely state what we want to learn thereby significantly reducing the data requirements. Furthermore, decomposing the transition model into several parts allows us to use reward shaping [2] in order to discourage policies based on dynamic characteristics.

Introducing constraints on the dynamic model based on abstract knowledge is an inherently problem-dependent process. This work explores this process for the heteroscedastic and bimodal Wet-Chicken benchmark [5, 6] which is both easy to understand and challenging to model. A central challenge in this benchmark is how to formulate a model which can represent bimodalities. One approach is presented in [7], where multimodal regression tasks are interpreted as a density estimation problem. A high number of candidate distributions is reweighed to match the observed data without modeling the underlying generative process. Reformulated in a Bayesian framework using latent variables, this approach has been applied to the Wet-Chicken benchmark in [8, 9]. However, such models are hard to interpret as they do not yield explicit models for the different modalities or their relative importance. In this work, we are interested in formulating a dynamics model which yields new interpretable insights about the underlying system. We formulate a probabilistic model which contains such explicit models by interpreting the Wet-Chicken benchmark as a data association problem [10, 11]. While many probabilistic interpretations of this problem assume that the relative importance of different modes is constant [12, 13], we base our formulation on the DAGP model [14] which learns a non-parametric model of each mode and where the associations between the modes themselves is further controlled by a non-stationary stochastic process.

The paper is outlined as follows. After introducing the Wet-Chicken benchmark, we show how high-level knowledge about this system can be used to impose Bayesian structure. We derive an efficient inference scheme for both the dynamics model and for probabilistic policy search based on variational inference. We show that this approach yields interpretable models and policies and is significantly more data-efficient than less interpretable alternatives.

2. The Wet-Chicken Benchmark

In the Wet-Chicken problem [5, 6], a canoeist is paddling in a two-dimensional river. The canoeist's position at time t is given by $\boldsymbol{s}_t = (x_t, y_t) \in \mathbb{R}^2$, where x_t denotes the position along the river and y_t the position across it. The river is bounded by its length $l = 5$ and width $w = 5$. There is a waterfall at the end of the river at $x = l$. The canoeist wants to get close to the waterfall to maximize the reward $r(\boldsymbol{s}_t) = r_t = x_t$. However, if the canoeist falls down the waterfall, he has to start over at the initial position $(0, 0)$.

The river's flow consists of a deterministic velocity $v_t = y_t \cdot {}^3\!/_w$ and stochastic turbulence $b_t = 3.5 - v_t$, both of which depend on the position on the y-axis. The higher y_t is, the faster the river flows but also the less turbulent it becomes. The canoeist chooses his paddle direction and intensity via an action $\boldsymbol{a}_t = (a_{t,x}, a_{t,y}) \in [-1, 1]^2$. The transition function $f : (\boldsymbol{s}_t, \boldsymbol{a}_t) \mapsto \boldsymbol{s}_{t+1} = (x_{t+1}, y_{t+1})$

is given by

$$x_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \\ 0 & \text{if } \hat{x}_{t+1} < 0 \\ \hat{x}_{t+1} & \text{otherwise} \end{cases} \quad y_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \text{ or } \hat{y}_{t+1} < 0 \\ w & \text{if } \hat{y}_{t+1} > w \\ \hat{y}_{t+1} & \text{otherwise} \end{cases} \tag{1}$$

where

$$\begin{aligned} \hat{x}_{t+1} &= x_t + (1.5 \cdot a_{t,x} - 0.5) + v_t + b_t \cdot \tau_t, \\ \hat{y}_{t+1} &= y_t + a_{t,y}, \end{aligned} \tag{2}$$

and $\tau_t \sim \mathcal{U}(-1, 1)$ is a uniform random variable that represents the turbulence.

There is almost no turbulence at $y = w$, but the velocity is too high to paddle back. Similarly, the velocity is zero at $y = 0$, but the canoeist can fall down the waterfall unpredictably due to the high turbulence. A successful canoeist must find a balance between handling the stochasticity and velocities within the capabilities of the canoeist to get as close to the waterfall as possible. However, as the canoeist moves closer to the waterfall, the distribution over the next states become increasingly more bi-modal as the probability of falling down increases. Together with the heteroscedasticity introduced by the turbulence dependent on the current position, these properties make the Wet-Chicken problem especially difficult for model-based reinforcement learning problems.

3. Probabilistic Policy Search

We are interested in finding a policy specified by the parameters $\boldsymbol{\theta}_\pi$ which maximizes the discounted return $J^\pi(\boldsymbol{\theta}_\pi) = \sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t) = \sum_{t=0}^{T} \gamma^t r_t$ with a constant discount factor $\gamma \in [0, 1]$. Starting from an initial state $\boldsymbol{s}_0$ we generate a trajectory of states $\boldsymbol{s}_0, \ldots, \boldsymbol{s}_T$ obtained by applying the action $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t)$ at every time step t. The next state is generated using the (latent) transition function f, yielding $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

Many environments have stochastic elements, such as the random drift in the Wet-Chicken benchmark from Section 2. We take this stochasticity into account by interpreting the problem from a Bayesian perspective where the discounted return specifies a generative model whose graphical model is shown in Fig. 1. Because of the Markov property assumed in RL, conditional independences between the states yield a recursive definition of the state probabilities given by

$$\begin{aligned} \mathrm{p}(\boldsymbol{s}_{t+1} \,|\, f, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(f(\boldsymbol{s}_t, \boldsymbol{a}_t) \,|\, \boldsymbol{s}_t, \boldsymbol{a}_t)\, \mathrm{p}(\boldsymbol{a}_t \,|\, \boldsymbol{s}_t, \boldsymbol{\theta}_\pi)\, \mathrm{p}(\boldsymbol{s}_t)\, \mathrm{d}\boldsymbol{a}_t\, \mathrm{d}\boldsymbol{s}_t, \\ \mathrm{p}(r_t \,|\, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(r(\boldsymbol{s}_t) \,|\, \boldsymbol{s}_t)\, \mathrm{p}(\boldsymbol{s}_t \,|\, \boldsymbol{\theta}_\pi)\, \mathrm{d}\boldsymbol{s}_t. \end{aligned} \tag{3}$$

With stochasticity or an uncertain transition model, the discounted return becomes uncertain and the goal can be reformulated to optimize the expected

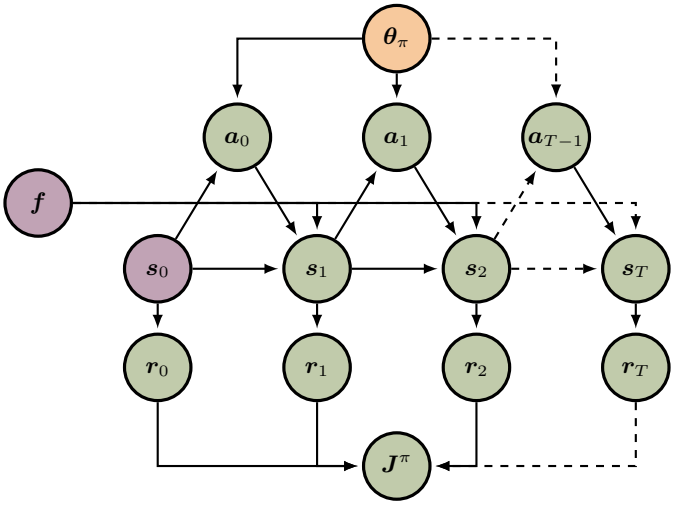


Figure 1: The generative process for the return J^π, where violet nodes are observed and parameters are shown in yellow. It shows how starting from $\boldsymbol{s}_0$, a trajectory of length T is generated with the policy parameterized by $\boldsymbol{\theta}_\pi$. The return is generated by the rewards which depend on their respective states only.

return

$$\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] = \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{p}(\boldsymbol{s}_t|\boldsymbol{\theta}_\pi)}[r_t]. \tag{4}$$

A model-based policy search method consists of two key parts [4]. First, a dynamics model is learned from state transition data. Second, this dynamics model is used to learn the parameters $\boldsymbol{\theta}_\pi$ of the policy π which maximize the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. We discuss both steps in the following.

3.1. An Interpretable Transition Model

We formulate a probabilistic transition model-based on high-level knowledge about the Wet-Chicken benchmark. Importantly, we do not formulate a specific parametric dynamics model as would be required to derive a controller. Instead, we make assumptions on a level typically available from domain experts.

We encode that given a pair of current state and action $\hat{\boldsymbol{s}}_t = (\boldsymbol{s}_t, \boldsymbol{a}_t)$, the next state $\boldsymbol{s}_{t+1}$ is generated via the combination of three things: the deterministic flow-behaviour of the river $\boldsymbol{f}_t$, some heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. This prior imposes structure which allows us to explicitly state what we want to learn from the data and where we do not assume prior knowledge: How does the river flow? What kind of turbulences exist? When does the canoeist fall down? How do the actions influence the system?

Each question is explicitly answered by one of the model's components. In Section 4 we will visualize these components and discuss how they can be used

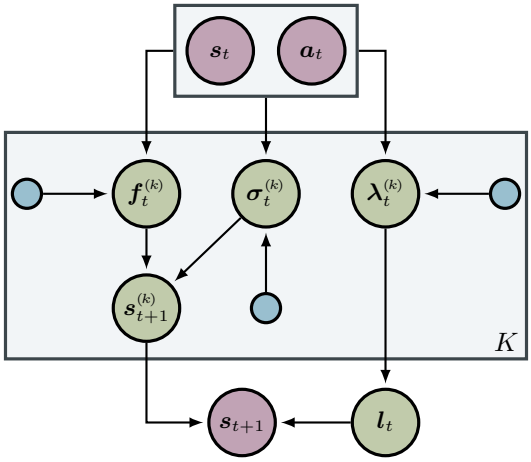


Figure 2: The graphical model for the DAGP-based transition model, where violet nodes are observed and variational parameters are blue. This model separates the flow-behaviour of the river $\boldsymbol{f}_t$, the heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. Latent variables $\boldsymbol{l}_t$ represent the belief that the t^{th} data point is a drop event.

by experts to gain new insights about the system. Additionally, interpretable transition models help to build trust in derived policies: Since experts can assess the plausibility of the transition model, successful policies are unlikely to behave unexpectedly on the true system.

We formulate a graphical model in Fig. 2 using the data association with GPs (DAGP) model [14], which allows us to handle the multi-modality introduced by falling down the waterfall. We specify this separation via the marginal likelihood

$$\mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \hat{\boldsymbol{s}}_t) = \int \mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \boldsymbol{f}_t, \boldsymbol{l}_t)\, \mathrm{p}(\boldsymbol{l}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{p}(\boldsymbol{\sigma}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{p}(\boldsymbol{f}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{d}\boldsymbol{\sigma}_t\, \mathrm{d}\boldsymbol{l}_t\, \mathrm{d}\boldsymbol{f}_t, \quad (5)$$

where $\boldsymbol{f}_t = \left(\boldsymbol{f}_t^{(1)}, \ldots, \boldsymbol{f}_t^{(K)}\right)$. The marginal likelihood consists of the two GPs $\mathrm{p}(\boldsymbol{\sigma}_t \,|\, \hat{\boldsymbol{s}}_t)$ and $\mathrm{p}(\boldsymbol{f}_t \,|\, \hat{\boldsymbol{s}}_t)$ and the two likelihoods

$$\begin{aligned} \mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \boldsymbol{f}_t, \boldsymbol{l}_t) &= \prod_{k=1}^{K} \mathcal{N}\Big(\boldsymbol{s}_{t+1} \,\Big|\, \boldsymbol{f}_t^{(k)}, \big(\boldsymbol{\sigma}_t^{(k)}\big)^2\Big)^{\mathbb{I}(l_t^{(k)}=1)}, \\ \mathrm{p}(\boldsymbol{l}_t \,|\, \hat{\boldsymbol{s}}_t) &= \int \mathcal{M}(\boldsymbol{l}_t \,|\, \mathrm{softmax}(\boldsymbol{\lambda}_t))\, \mathrm{p}(\boldsymbol{\lambda}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{d}\boldsymbol{\lambda}_t \end{aligned} \quad (6)$$

where $\mathcal{M}$ denotes a multinomial distribution. These likelihood describe the regression and classification tasks implied by the problem respectively: In our case, we use $K = 2$ modes, one for staying in the river and one for falling down the waterfall. For every data point we infer a posterior belief $\mathrm{p}(\boldsymbol{l}_t)$ about which mode the data point belongs to, as we assume this separation can not be predetermined using expert knowledge.

We place independent GP priors on the $\boldsymbol{f}^{(k)}$, $\boldsymbol{\sigma}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$. Given the data a fixed set of assignments $\boldsymbol{L}$, our modelling assumptions imply independence

between the K modes. However, this independence is lost if the assignments are unknown and a discrete optimization problem has to be solved when doing joint inference over the different modes and the association problem. We approximate the exact posterior via a factorized variational distribution

$$\mathrm{q}(\boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \boldsymbol{U}) = \prod_{k=1}^{K} \prod_{t=1}^{T} \mathrm{q}(\boldsymbol{f}_t^{(k)}, \boldsymbol{u}^{(k)}) \, \mathrm{q}(\boldsymbol{\lambda}_t^{(k)}, \boldsymbol{u}_{\boldsymbol{\lambda}}^{(k)}) \, \mathrm{q}(\boldsymbol{\sigma}_t^{(k)}, \boldsymbol{u}_{\boldsymbol{\sigma}}^{(k)}) \tag{7}$$

which introduces variational inducing inputs and outputs $\boldsymbol{U}$ as described in [14–16]. These inducing inputs independently characterize the respective model parts and enable us to do inference via stochastic optimization.

The variational parameters are optimized by minimizing a lower bound to the marginal likelihood which can be efficiently computed via sampling and enables stochastic optimization.

$$\begin{aligned}
\mathcal{L}_{\mathrm{DAGP}} &= \mathbb{E}_{\mathrm{q}(\boldsymbol{F}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \boldsymbol{U})} \left[\log \frac{\mathrm{p}\Big(\boldsymbol{S}', \boldsymbol{F}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \boldsymbol{U} \,\Big|\, \hat{\boldsymbol{S}}\Big)}{\mathrm{q}(\boldsymbol{F}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \boldsymbol{U})} \right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\mathrm{q}(\boldsymbol{f}_t)}[\log \mathrm{p}(\boldsymbol{s}_t' \,|\, \boldsymbol{f}_t, \boldsymbol{\lambda}_t, \boldsymbol{\sigma}_t)] + \sum_{t=1}^{T} \mathbb{E}_{\mathrm{q}(\boldsymbol{\lambda}_t)}[\log \mathrm{p}(\boldsymbol{l}_t \,|\, \boldsymbol{\lambda}_t)] \\
&\quad - \sum_{k=1}^{K} \mathrm{KL}(\mathrm{q}\big(\boldsymbol{u}^{(k)}, \boldsymbol{u}_{\boldsymbol{\lambda}}^{(k)}, \boldsymbol{u}_{\boldsymbol{\sigma}}^{(k)}\big) \parallel \mathrm{p}\big(\boldsymbol{u}^{(k)}, \boldsymbol{u}_{\boldsymbol{\lambda}}^{(k)}, \boldsymbol{u}_{\boldsymbol{\sigma}}^{(k)}\big))
\end{aligned} \tag{8}$$

To gain informative gradients with respect to the assignments $\boldsymbol{l}$ and assignment process $\boldsymbol{\lambda}$, we use a continuous relaxation based on Concrete random variables [17]. We represent the belief about $\boldsymbol{l}_t$ as a K-dimensional discrete distribution $\mathrm{q}(\boldsymbol{l}_t)$. Instead of drawing discrete samples from $\mathrm{q}(\boldsymbol{l}_t)$ when calculating $\mathcal{L}_{\mathrm{DAGP}}$, we draw samples $\hat{\boldsymbol{l}}_t$ from a concrete random variable. Based on a temperature parameter, concrete random variables yield samples which are almost discrete but which still yield informative gradients. For details we refer to [14].

We obtain an explicit representation of the GP posteriors during variational inference which allows us to efficiently propagate samples through the model to simulate trajectories used for policy search. Predictions for an unknown state $\hat{\boldsymbol{s}}_*$ are mixtures of K independent Gaussians given by,

$$\begin{aligned}
\mathrm{q}(\boldsymbol{s}_*' \,|\, \hat{\boldsymbol{s}}_*) &= \int \sum_{k=1}^{K} \mathrm{q}(l_*^{(k)} \,|\, \hat{\boldsymbol{s}}_*) \, \mathrm{q}(\boldsymbol{s}_*'^{(k)} \,|\, \hat{\boldsymbol{s}}_*) \, \mathrm{d}\boldsymbol{l}_* \\
&= \int \sum_{k=1}^{K} \mathrm{q}(l_*^{(k)} \,|\, \hat{\boldsymbol{s}}_*) \, \mathrm{q}(\boldsymbol{s}_*'^{(k)} \,|\, \boldsymbol{f}_*^{(k)} \boldsymbol{\sigma}_*^{(k)}) \, \mathrm{q}(\boldsymbol{f}_*^{(k)}, \boldsymbol{\sigma}_*^{(k)} \,|\, \hat{\boldsymbol{s}}_*) \, \mathrm{d}\boldsymbol{l}_* \, \mathrm{d}\boldsymbol{f}_* \, \mathrm{d}\boldsymbol{\sigma}_* \\
&\approx \sum_{k=1}^{K} \tilde{l}_*^{(k)} \, \tilde{\boldsymbol{s}}_*'^{(k)}.
\end{aligned} \tag{9}$$

We sample from the assignment process $\boldsymbol{l}_*$ and heteroscedastic noise process $\boldsymbol{\sigma}_*$. The K predictive posteriors $\mathrm{q}(\boldsymbol{s}_*'^{(k)} \,|\, \boldsymbol{f}_*^{(k)} \boldsymbol{\sigma}_*^{(k)})$ are then given by K independent shallow GPs and can be computed analytically.

3.2. Policy Learning

After training a transition model, we use the variational posterior $\mathrm{q}(\boldsymbol{s}_{t+1} \,|\, \hat{\boldsymbol{s}}_t)$ to train a policy by sampling roll-outs and optimizing policy parameters via stochastic gradient descent on the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. The expected return is approximated using the variational posterior given by

$$\begin{aligned}
\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] &= \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{p}(\boldsymbol{s}_t|\boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t] \approx \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{q}(\boldsymbol{s}_t|\boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t] \\
&= \int \sum_{t=0}^{T} \left[\gamma^t \, \mathbb{E}_{\mathrm{q}(\boldsymbol{s}_t|\boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t] \right] \mathrm{p}(\boldsymbol{s_0}) \prod_{t=0}^{T-1} \mathrm{q}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{s}_t, \boldsymbol{\theta}_\pi) \, \mathrm{d}\boldsymbol{s}_0 \ldots \mathrm{d}\boldsymbol{s}_T \quad (10) \\
&\approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t r_t^p .
\end{aligned}$$

We expand the expectation to explicitly show the marginalization of the states in the trajectory. Due to the Markovian property of the transition dynamics, the integral factorizes along t. The integral is approximated by averaging over P samples propagated through the model starting from a known distribution of initial states $\mathrm{p}(\boldsymbol{s}_0)$. State transitions can be efficiently sampled from the variational posterior of the dynamics model by repeatedly taking independent samples of the different GPs.

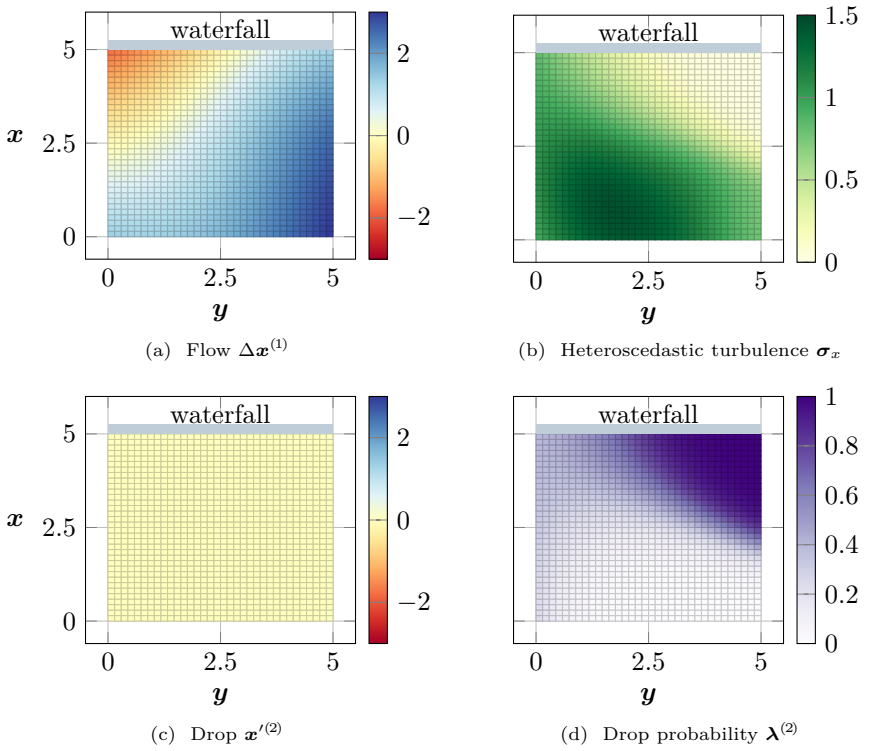
The expected return in (10) can be optimized using stochastic gradient descent via the gradients

$$\nabla_{\theta_\pi} J^\pi(\theta_\pi) \approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t \nabla_{\theta_\pi} r_t^p \quad (11)$$

of the Monte Carlo approximation as they are an unbiased estimator of the true gradient. The gradients of the samples can be obtained using automatic differentiation tools such as TensorFlow [18]. The P roll-outs can be trivially parallelized. Importantly, we only need a small number of Monte Carlo samples at every iteration, since we use the gradients of the samples directly.

4. Results

To solve the Wet-Chicken problem, we first train the dynamics model on batch data sampled from the true dynamics and then optimize neural policies with respect to this dynamics model. As the DAGP-based dynamics model is designed to be interpretable, we first discuss how, additionally to a joint posterior, the independent posteriors of its components yield insights about



(a) Flow $\Delta \boldsymbol{x}^{(1)}$

(b) Heteroscedastic turbulence $\boldsymbol{\sigma}_x$

(c) Drop $\boldsymbol{x}'^{(2)}$

(d) Drop probability $\boldsymbol{\lambda}^{(2)}$

Figure 3: The separation of different aspects of the Wet-Chicken in DAGP-based transition models benchmark yields new and interpretable information about the underlying dynamics. The different parts of the model explicitly show flow speeds (Fig. 3a), turbulence (Fig. 3b), drop behaviour (Fig. 3c) and drop probabilities (Fig. 3d) with respect to the current position in the river and action $a = 0$. The model has learned that the river is turbulent on the left and fast on the right, leading to consistent medium drop probabilities on the left due to stochasticity and a sharp boundary on the right, where the flow speed dominates. Note that a drop resets the position to the initial state irrespective of the current state. It is therefore correctly learned to be represented by the constant zero function (Fig. 3c).

the Wet-Chicken problem. We then show how successful policies can be found with less data compared to the model-free NFQ [19] and model-based Bayesian Neural Networks with latent variables (BNN+LV) [8], two approaches which do not makes use of high-level expert knowledge. Thirdly, we show how the human-interpretable components of the dynamics models can be used for reward shaping, allowing us to easily formulate a requirement for conservative policies.

4.1. Dynamics Model

The benchmark has two-dimensional state and action spaces from which we sample uniform random transitions with varying N in the range 100 to 5000. For $N \geq 250$, our model is able to identify the underlying dynamics. In Fig. 4 we show the joint predictive posterior of a DAGP-based transition model. The

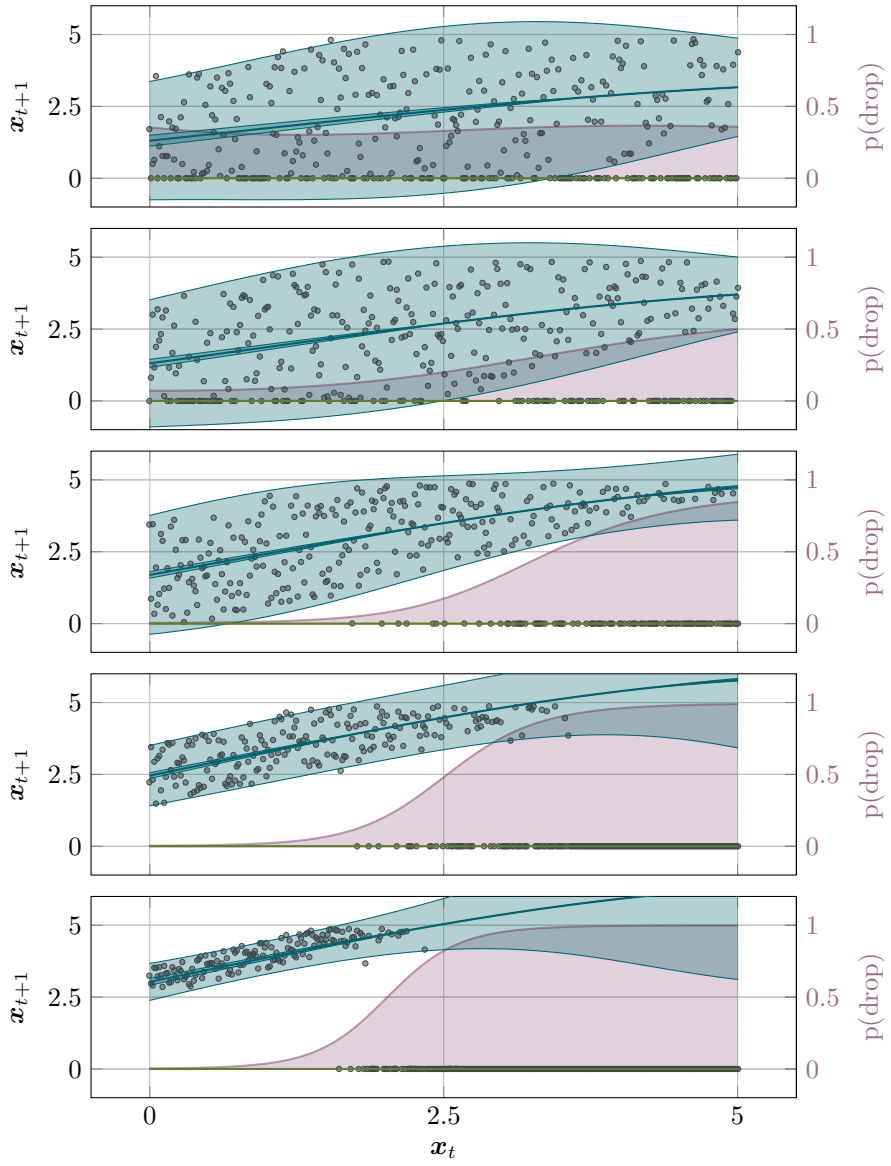


Figure 4: Linear cuts through the DAGP-based transition model with the waterfall on the right at $x_t = 5$. The plots show the dependency between x_t and x_{t+1} with respect to the action $a_t = 0$ and, from top to bottom, $y_t \in \{0, 1, 2.5, 4, 5\}$. The DAGP-based transition model successfully separates the two modes introduced through flow (blue) and drop (green) behaviours and predicts the probability of being assigned to the drop mode (violet). While drops can be modelled using a constant noiseless function, the flow speed (gradient and bias) and heteroscedastic noise (variance) varies in the different cuts. For low y_t, the river flows slowly but is very turbulent, while for high y_t, the river flows fast but deterministically.

different plots show linear cuts through the Wet-Chicken system with respect to the action $a_t = 0$ and $y_t \in \{0, 1, 2.5, 4, 5\}$. The transition model has successfully identified the two modes introduced through flow and drop behaviours and their relative importance. These cuts require additional examination to recover new knowledge about the system's behaviour. In contrast, the separation of the learning problem in the DAGP-based dynamics model gives us explicit and separate posteriors about the different system components via the independent GP posteriors shown in Fig. 3. This belief can directly be reasoned about with experts to evaluate the environment in which policies will be trained, raising confidence in their correctness.

While drops can be modelled using a constant noiseless function, the flow speed and heteroscedastic turbulence varies throughout the system. For low y the river flows slowly but is very turbulent while for high y the river flows fast but deterministically. In the turbulent regime, falling down is possible but not certain for most x, while in the flow dominated regime, a drop becomes highly probable under a certain distance from the waterfall. Note that even though the turbulence as defined in Section 2 is independent of x, the heteroscedastic noise process has uncovered the implicit dependency for high x as most possible turbulence values lead to falling down and thus assignment to the other mode. Similarly, the flow speed shown in Fig. 3a is negative in the top left corner. This is due to the fact that the flow mode models the position after one step under the condition of not falling. As most turbulence into the direction of the waterfall leads to a drop, the posterior mean is further away from the waterfall as the turbulence dominates the low flow speed on the left side of the river.

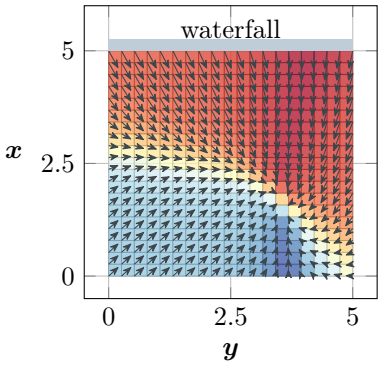
4.2. Policy Learning

Given a posterior for the dynamics model, we now train a neural policy using probabilistic model rollouts. We sample initial states from the training data, use a horizon of $T = 5$ steps and average over $P = 20$ samples with $\gamma = 0.9$. We use a two-layer neural network with 20 ReLU-activated units each as our policy parametrization. For every state transition, we sample independently from the different model components to generate a sample for the next state using (9). This incorporates both the stochasticity in the system introduced via heteroscedastic noise and the Bayesian uncertainty about the correct model in the policy search. During training, the policy thus implicitly learns to consider the stochasticity of the Wet-Chicken benchmark as different sample-trajectories generate gradients with respect to different realizations of the stochasticity of the Wet-Chicken benchmark. Figure 5b shows how a successful policy has found a trade-off between the unpredictability on the left and the uncontrollable speed on the right.

In Table 5a, we compare policy search based on the DAGP-based dynamics model with a standard GP dynamics model and NFQ. We present expected returns for training runs with different amounts of data averaged over 10 experiments together with standard errors. A policy applying uniformly random actions yields a return of about 1.5 and a return above 2.2 indicates that a successful policy has been found. We ran NFQ for 20 full model learning and

N	NFQ	BNN+LV	GP	DAGP
100	0.66 ± 0.16	—	$\mathbf{1.41 \pm 0.01}$	1.18 ± 0.09
250	1.71 ± 0.07	1.62 ± 0.20	1.54 ± 0.01	$\mathbf{2.33 \pm 0.01}$
500	1.60 ± 0.10	2.18 ± 0.07	1.56 ± 0.01	$\mathbf{2.25 \pm 0.01}$
1000	1.99 ± 0.06	2.27 ± 0.01	2.13 ± 0.01	$\mathbf{2.32 \pm 0.01}$
2500	2.26 ± 0.02	$\mathbf{2.30 \pm 0.01}$	1.91 ± 0.01	2.28 ± 0.01
5000	$\mathbf{2.33 \pm 0.01}$	2.30 ± 0.01	1.91 ± 0.01	2.28 ± 0.01

(a) Comparison of expected returns



(b) A successful Wet-Chicken policy

Figure 5: Using interpretable DAGP-based transition models with structurally informative priors, successful policies can be learned based on 250 observations. In contrast, about 2500 observations are needed to find a policy using the model-free NFQ. As GP based transition models are not capable of representing bimodal dynamics, training does not yield successful policies. The chosen optimal movement direction of a successful policy (right) is denoted by both the arrows and background color.

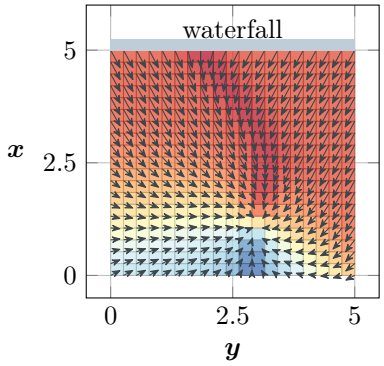
sampling iterations using a neural network with one 10-unit hidden layer with sigmoid activations.

A standard GP cannot model heteroscedastic noise or multi-modality. For any point in the input space, the GP can therefore only predict that the agent will always fall down, never fall down or, via very high uncertainties, that any state in system is possible. None of these possibilities represent the dynamics well enough to allow the policy search to derive a policy, illustrating our need for a more structured model. For $N \geq 250$, the DAGP-based dynamics model identifies the underlying dynamics well and policies can be found reliably.

BNN+LV is a more expressive model that can represent both heteroscedasticity and multi-modality. Due to the model's structure however, it is hard to incorporate high-level expert knowledge and therefore, more sturcture has to be learned from the data. BNN+LV reliably finds good policies for $N \geq 1000$ and sometimes finds good policies for $N = 500$. As this approach is model-based and formulates a reasonable general-purpose prior on the wfor the dynamics, the results fall between the more informed DAGP, which is successful with less data,

Training Reward	Original Return	Conservative Return	Drop %
r_{orig}	$\mathbf{2.32 \pm 0.01}$	-1.22 ± 0.02	21.8 ± 0.2
r_{cons}	2.17 ± 0.01	$\mathbf{-1.00 \pm 0.01}$	$\mathbf{18.9 \pm 0.1}$

(a) Drop risk reduction with reward shaping



(b) A conservative Wet-Chicken policy

Figure 6: As the different components of a DAGP-based transition model are easily interpretable, they can be used for reward shaping. We formulate a conservative reward function r_{cons} which penalizes drops and can easily be evaluated in the transition model. A resulting policy (right) has worse return with respect to the original reward function r_{orig} but effectively reduces the risk of falling.

and NFQ, which is more uninformed.

NFQ is a model-free approach. Instead of learning a dynamics model and using rollouts in that model to find a good policy, NFQ directly models the optimal Q-function and thus the optimal policy. A Q-function represents the expected return after taking a specified action in a specified state. Since the expected return already takes into account both the heteroscedasticity and multimodality of the system, the Q-function itself can be modelled with a standard function approximator such as a neural network. Thus, no special modelling is needed when applying NFQ to the Wet-Chicken benchmark and given enough data, NFQ is able to find successful policies. However, at the same time, not modelling the dynamics explicitly also prevents us from utilising the high-level expert knowledge we have about the system, thus increasing the required amount of data: Using DAGP-based dynamics models, a successful policy can be found with about an order of magnitude less data.

4.3. Reward Shaping

We have shown how a dynamics model informed by high-level expert knowledge increases data efficiency. A further advantage of the decomposition of the dynamics model in interpretable components is that the predictions of these

	DAGP				
N	$K = 1$	$\boldsymbol{K = 2}$	$K = 3$	$K = 4$	$K = 5$
250	1.41 ± 0.01	$\mathbf{2.33 \pm 0.01}$	1.64 ± 0.38	1.31 ± 0.25	1.65 ± 0.08
500	1.54 ± 0.01	$\mathbf{2.25 \pm 0.01}$	1.97 ± 0.23	1.48 ± 0.21	2.14 ± 0.10
1000	2.13 ± 0.01	$\mathbf{2.32 \pm 0.01}$	1.99 ± 0.17	2.09 ± 0.12	2.16 ± 0.09
2500	1.91 ± 0.01	$\mathbf{2.28 \pm 0.01}$	2.15 ± 0.03	2.06 ± 0.12	2.17 ± 0.03
5000	1.91 ± 0.01	$\mathbf{2.28 \pm 0.01}$	2.19 ± 0.06	1.95 ± 0.16	2.08 ± 0.13

Table 1: Comparison of expected returns for different settings of K

components can be used to influence the policy search. In this example, we want to find a more conservative policy which, when compared to Fig. 5b, sacrifices some return in order to avoid falling down the waterfall.

Any successful agent has the implicit incentive to avoid drops as they move the canoeist away from the waterfall. However, a successful policy still accepts that it will fall down sometimes due to turbulence. To encourage more conservative behaviour, we use a conservative reward

$$r_{\text{cons}}(\boldsymbol{s}) = r_{\text{orig}}(\boldsymbol{s}) \cdot (1 - \text{p}(\text{drop} \,|\, \boldsymbol{s})) - 5 \cdot \text{p}(\text{drop} \,|\, \boldsymbol{s}) \tag{12}$$

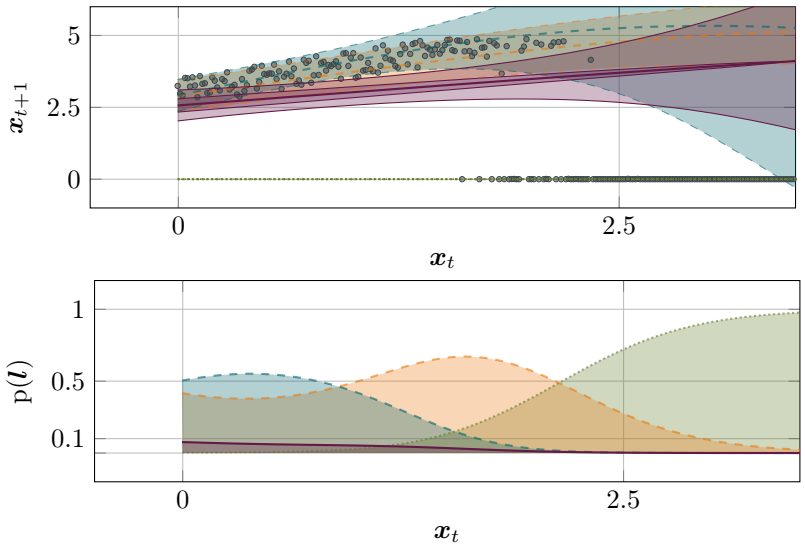
which includes the original Wet-Chicken reward function $r_{\text{orig}}((x, y)) = x$. For every state, the DAGP-based dynamics model yields an explicit drop-probability which can easily be evaluated. The conservative reward punishes a high drop probability reweighed with the maximum original reward $\max_{\boldsymbol{s}} r_{\text{orig}}(\boldsymbol{s}) = 5$.

Figure 6 shows a resulting conservative policy. Such a policy avoids both the turbulent states on the left and the fast flowing states on the right. It tries to reach a sweet spot, which, compared to Fig. 5b, is further away from the waterfall and therefore safer. We compare 10 runs with $N = 1000$ observations using the original reward and the conservative reward. The resulting conservative policies yield lower return than the more aggressive default policies but reliably reduce drop probabilities as well. The interpretable nature of the dynamics models have allowed us to easily influence policy behaviours.

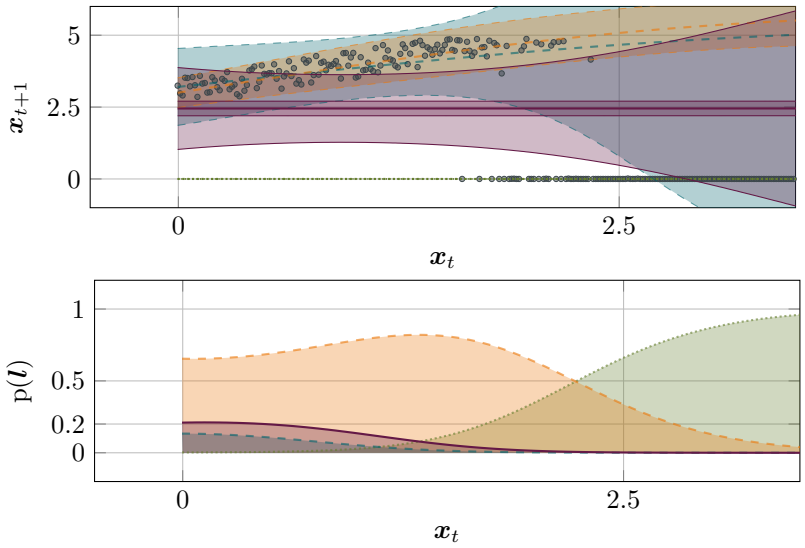
4.4. Effects of Model Misspecification

In Section 3 we have formulated a dynamics model informed by high-level expert knowledge. One important insight we assumed is the bimodal nature of the Wet-Chicken problem introduced by the waterfall. In Section 4.2, we compared our model to standard GPs and showed that modelling multi-modality is critical to solve Wet-Chicken. We extend this comparison in this experiment and discuss the effects of model misspecification on our model's performance. Specifically, we investigate the case where additional modes are available to dynamics model to solve the underlying data association problem.

Table 1 shows results for $K \in \{1, \ldots, 5\}$, where $K = 1$ is equivalent to standard GPs. All models have been trained for the same number of iterations



(a) Transition model with $K = 4$ and successful policy training



(b) Transition model with $K = 4$ and failed policy training

Figure 7: Comparison of linear cuts through two DAGP-based transition models with $K = 4$ at $y_t = 5$ and $a_t = 0$. The respective upper plots show the predictive posterior of the different modes while the lower plots show assignment probabilities to the different modes. For both models, one mode (green, dotted) model drops and two modes (blue and yellow, dashed) represent flow behaviour. A third more uninformed mode (violet) is almost irrelevant in the first model but explains some data through a high noise variance in the second model. A significant amount of predictions from the second model are uninformed, leading to the failure of the policy search.

and, for $K > 1$, all models have comparable marginal likelihoods. While 250 data points are enough with $K = 2$ to reliably solve the Wet-Chicken problem, more data is needed until working policies can be found with $K > 2$ (e.g. for $K = 5$, double the data was needed until one of the runs found a working policy). Most notably, performance fluctuates significantly with misspecified models for different repetitions of the same experiment and good policies can not be found reliably.

Using the additional modes available, the model can now find representations of the systems where multiple modes jointly represent the river's flow. This showcases how data association problems are inherently ill-posed in general [10, 11]. The additional representative power for $K > 2$ introduces symmetries in the optimization landscape which both significantly complicate training [12, 20] and lead to undesired generalization behaviour which is not driven by knowledge about the underlying system.

An example for undesired generalization is shown in Fig. 7 which compares two models trained with $K = 4$ and $N = 2500$. While both models explain the overall training data well, the cuts through the system at $y_t = 5$ give an intuition why the first model leads to a successful policy, while the second model does not. Both models represent drops via one of the modes and share the remaining three modes to jointly explain the flow behaviour. In the first model, two alternating modes have learned essentially equivalent models and a third more uninformed mode is almost irrelevant. The second model is similar, but the uninformed mode's model is closer to the RBF prior and more relevant. Note that due to the high noise variance, this choice of model still explains the data only slightly worse. Still, the second model does not generalize according to the underlying system. A significant amount of predictions from the second model are uninformed, leading to the failure of the policy search.

Significantly longer training or specialized optimization schemes may lead to robust inference for $K > 2$. However, this experiment shows the significance of encoding available abstract prior knowledge to avoid pathologic model behaviours. Models for $K = 2$ both reliably identify the system using standard optimization methods and yield immediately interpretable results.

5. Conclusion and Discussion

In this paper we have presented a Bayesian reinforcement learning model-based on non-parametric Gaussian process priors. The model is motivated by the observation that in real world scenarios high-level prior knowledge of the system dynamics is often available. We believe that many tasks are characterised by dynamics that can be decomposed into several attributes. For example, when a physical structure is excited by a force oscillating at its natural frequency its response will change drastically. The approach we have presented is based on learning a modular dynamic model which decomposes this type of transitional behaviour into separate components. The model learns both the individual components and the underlying structure of how the components interact within

the system. The use of Gaussian process priors to quantify the uncertainty within components allows us to perform probabilistic policy search.

The interpretable structure of our model facilitates data efficient learning by easily incorporating prior knowledge. We showed experimentally how this significantly reduces the data requirements compared to a model free approach. Furthermore, the same knowledge can be used as a means for directing the policy search by discouraging solutions which exhibit a specific dynamic, such as avoiding falling down the waterfall in the Wet-Chicken benchmark.

References

[1] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, New York, NY, USA, 2014.

[2] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, 1998.

[3] S. Lange, T. Gabel, M. Riedmiller, Batch reinforcement learning, in: Reinforcement Learning, Springer, 2012, pp. 45–73.

[4] M. Deisenroth, C. E. Rasmussen, PILCO: A model-based and data-efficient approach to policy search, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 465–472.

[5] V. Tresp, The wet game of chicken, Siemens AG, CT IC 4, Technical Report (1994).

[6] A. Hans, S. Udluft, Efficient uncertainty propagation for reinforcement learning with limited data, in: International Conference on Artificial Neural Networks, Springer, 2009, pp. 70–79.

[7] C. M. Bishop, Mixture Density Networks, Technical Report, 1994.

[8] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, S. Udluft, Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks, arXiv:1605.07127 [cs, stat] (2016). `arXiv:1605.07127`.

[9] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, S. Udluft, Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning, in: International Conference on Machine Learning, 2018, pp. 1192–1201.

[10] Y. Bar-Shalom, Tracking and Data Association, Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[11] I. J. Cox, A review of statistical data association techniques for motion correspondence, International Journal of Computer Vision 10 (1993) 53–66.

[12] M. Lázaro-Gredilla, S. Van Vaerenbergh, N. D. Lawrence, Overlapping mixtures of Gaussian processes for the data association problem, Pattern Recognition 45 (2012) 1386–1395.

[13] E. Bodin, N. D. F. Campbell, C. H. Ek, Latent Gaussian Process Regression, arXiv:1707.05534 [cs, stat] (2017). arXiv:1707.05534.

[14] M. Kaiser, C. Otte, T. Runkler, C. H. Ek, Data Association with Gaussian Processes, arXiv:1810.07158 [cs, stat] (2018). arXiv:1810.07158.

[15] A. Damianou, N. Lawrence, Deep Gaussian Processes, in: Artificial Intelligence and Statistics, 2013, pp. 207–215.

[16] J. Hensman, A. G. d. G. Matthews, Z. Ghahramani, Scalable variational Gaussian process classification, Journal of Machine Learning Research 38 (2015) 351–360.

[17] C. J. Maddison, A. Mnih, Y. W. Teh, The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables, arXiv:1611.00712 [cs, stat] (2016). arXiv:1611.00712.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015). Software available from tensorflow.org.

[19] M. Riedmiller, Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method, in: European Conference on Machine Learning, Springer, 2005, pp. 317–328.

[20] T. P. Minka, Expectation Propagation for approximate Bayesian inference, in: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2001, pp. 362–369. arXiv:1301.2294.